

ipd1100madbvpmTES-10

**Defense Information Infrastructure (DII)
Common Operating Environment (COE)
Programming Manual (PM)
for the
Digitized Bathymetry API Segment (MADBV)
of the
Tactical Environmental Support System Next Century
[TESS(NC)]
Meteorology and Oceanography (METOC) Database
Preliminary Release**

Document Version 1.1

2 October 1998

**Prepared for:
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA**

**Prepared by:
Integrated Performance Decisions, Inc.
Middletown, RI**

PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE
ipd1100madbvpmsTES-10

Table of Contents

1	SCOPE	1
1.1	Identification	1
1.2	System Overview	1
2	REFERENCED DOCUMENTS	5
2.1	Government Documents.....	5
2.2	Non-Government Documents.....	6
3	SEGMENT OVERVIEW	7
4	SEGMENT DEVELOPMENT	9
4.1	Writing Applications Using the MADBV APIs.....	9
4.1.1	Retrieving Available Database Resolutions.....	10
4.1.2	Retrieving Bathymetry Along a Track.....	12
4.1.3	Retrieving Gridded Bathymetry.....	17
4.1.4	Retrieve Bathymetry Contours.....	22
4.2	MDDBV Database Overview	26
4.3	Building Applications Using the MADBV Libraries.....	27
4.3.1	Makefile Using the Dynamic/Static MADBV Libraries on HP-UX	27
4.3.2	Makefile Using the Dynamic MADBV Libraries on Windows NT	28
4.3.3	Makefile Using the Static MADBV Libraries on Windows NT	28
5	CUSTOMIZING SEGMENTS	31
6	NOTES.....	33
6.1	Glossary of Acronyms.....	33

List of Figures

1-1 TESS(NC) METOC Database - DII COE Segment View 3
1-2 Distributed APIs via COTS RDMBS Client/Server Functionality 3
1-3 Distributed APIs via DII COE Kernel Services (NFS)..... 4

1 SCOPE

1.1 Identification

This Programming Manual (PM) describes the use of the Digitized Bathymetry Application Program Interface (API) Segment (MADBV), Version 1.1, of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database. The MADBV segment provides APIs for the retrieval of historical bathymetry data. This software is designed to run under the Defense Information Infrastructure (DII) Common Operating Environment (COE), release 3.1, on a Hewlett-Packard computer running HP-UX 10.20 or a personal computer running the Microsoft Windows NT 4.0 operating system with Service Pack 3.

1.2 System Overview

The APIs described in this document form a portion of the METOC Database component of the TESS(NC) Program (Navy Integrated Tactical Environmental Subsystem (NITES) Version I). On 29 October 1996, the Oceanographer of the Navy issued a TESS Program Policy statement in letter 3140 Serial 961/6U570953, modifying the Program by calling for five seamless software versions that are DII COE compliant, preferably to level 5.

The five versions are:

- NITES Version I The local data fusion center and principal METOC analysis and forecast system (TESS(NC))
- NITES Version II The subsystem on the Joint Maritime Command Information System (JMCIS) or Global Command and Control System (GCCS) (NITES/Joint METOC Segment (JMS))
- NITES Version III The unclassified aviation forecast, briefing and display subsystem tailored to Naval METOC shore activities (currently satisfied by the Meteorological Integrated Data Display System (MIDDS))
- NITES Version IV The Portable subsystem composed of independent PCs/workstations and modules for forecaster, satellite, communications, and Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance (IC4ISR) functions (currently the Interim Mobile Oceanographic Support System (IMOSS))

- NITES Version V Foreign Military Sales (currently satisfied by the Allied Environmental Support System (AESS))

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps weather forecasters and tactical planners. NITES I provides these users METOC data, products, and applications necessary to support the warfighter in tactical operations and decision making. NITES I provides METOC data and products to NITES I and NITES II applications, as well as non-TESS(NC) systems requiring METOC data, in a heterogeneous networked computing environment.

The TESS(NC) Concept of Operations and system architecture require that the METOC Database be distributed both in terms of application access to METOC data and products and in terms of physical location of the data repositories. The organizational structure of the database is influenced by these requirements, and the components of this distributed database are described below.

In accordance with DII COE database concepts, the METOC Database is currently composed of five DII COE compliant *shared database* segments and one DII COE compliant data segment. Associated with each shared database and data segment is an API segment. This organization is shown in Figure 1-1. The segments are arranged by data type as follows:

<u>Data Type</u>	<u>Data Segment</u>	<u>API Segment</u>
Grid Fields	MDGRID	MAGRID
Latitude-Longitude-Time (LLT) Observations	MDLLT	MALLT
Textual Observations and Bulletins	MDTXT	MATXT
Remotely Sensed Data	MDREM	MAREM
Imagery	MDIMG	MAIMG
Historic Bathymetry Data	MDDBV	MADBV

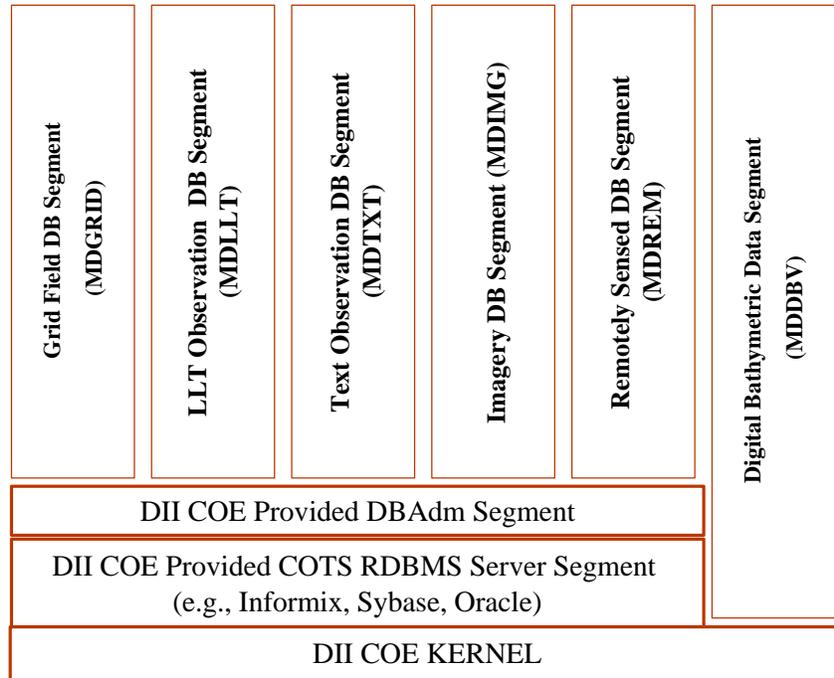


Figure 1-1. TESS(NC) METOC Database - DII COE Segment View

Typical client-server installations access shared database segments via a COTS RDBMS client/server as shown in Figure 1-2. This shows the shared database segments residing on a DII COE SHADE database server, with a NITES I or II client machine hosting the API segments. Communication between API segments and shared database segments is accomplished over the network using ANSI-standard Structured Query Language (SQL).

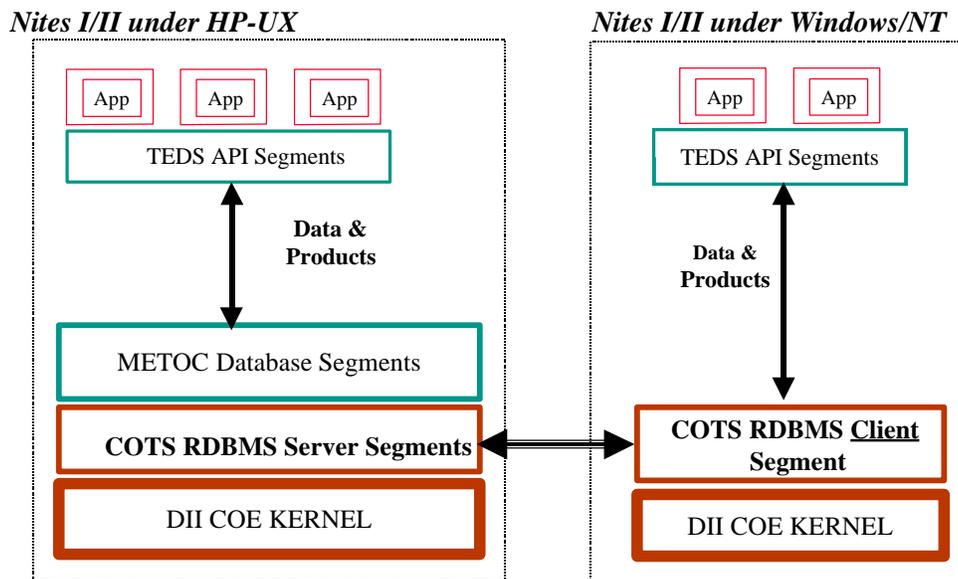


Figure 1-2. Distributed APIs via COTS RDMBS Client/Server Functionality

Data Segments are static files of historic data. DII COE data segments are available over a distributed network via DII COE Kernel Service (NFS). In this case, the data segments are accessed directly by the distributed APIs (Figure 1-3). The platform running the applications needing the data must first mount the file system containing the data segment. The remote system may then access the data from the mounted drive using NFS services. Access to the mounted drive is then transparent to the application/API utilizing the data.

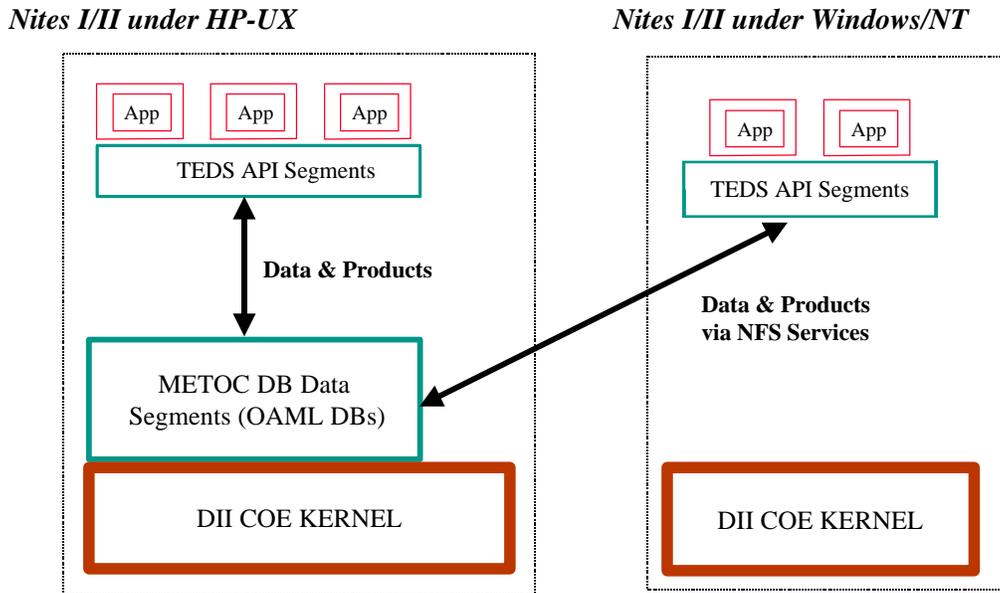


Figure 1-3. Distributed APIs via DII COE Kernel Services (NFS)

The MADBV segment deals with historic bathymetry data. The data is generated from the Oceanographic and Atmospheric Master Library (OAML) Digitized Bathymetric Database – Variable resolution (DBDB-V) data, and provides global water depth at various resolutions throughout the world.

2 REFERENCED DOCUMENTS

2.1 Government Documents

STANDARDS

MIL-STD-498 *Software Development and Documentation*
5 December 1994

SPECIFICATIONS

Unnumbered *Performance Specification (PS) for the Tactical*
5 December 1997 *Environmental Support System/Next Century TESS(3)/NC*
(AN/UMK-3)

Unnumbered *Software Requirements Specification for the Tactical*
30 September 1997 *Environmental Support System/Next Century [TESS(3)/NC]*
Meteorological and Oceanographic (METOC) Database,
Space and Naval Warfare Systems Command, Environmental
Systems Program Office (SPAWAR PMW-185), Washington,
DC

OTHER DOCUMENTS

Unnumbered *Database Design Description for the Tactical Environmental*
30 September 1997 *Support System/Next Century [TESS(3)/NC] Meteorological*
and Oceanographic (METOC) Database, Space and Naval
Warfare Systems Command, Environmental Systems Program
Office (SPAWAR PMW-185), Washington, DC

DII.COE.DocReqs-5 *Defense Information Infrastructure (DII) Common Operating*
29 April 1997 *Environment (COE) Developer Documentation*
Requirements, Version 1.0

DII.COE31.HP10.20.CIP *DII COE V3.1 HP 10.20 Consolidated Installation*
23 May 1997 *Procedures*

- DII.3010.HP1020.KernelP1.IG-1 *DII COE Kernel 3.0.1.OP1 Patch 1 for HP-UX 10.20*
9 May 1997 *Installation Guide*
- DII.3010.HP1020.KernelP2.IG-1 *DII COE Kernel 3.0.1.OP2 Patch 2 for HP-UX 10.20*
30 July 1997 *Installation Guide*
- DII.3010.HP1020.KernelP3.IG-1 *DII COE Kernel 3.0.1.OP3 Patch 3 for HP-UX 10.20*
08 August 1997 *Installation Guide*
- DII.3010.HP1020.KernelP4.IG-1 *DII COE Kernel 3.0.1.OP4 Patch 4 for HP-UX 10.20*
27 August 1997 *Installation Guide*
- Unnumbered *Application Program Interface Reference Manual (APIRM)*
2 October 1998 *for the Digitized Bathymetric Database API Segment*
 (MADBV) of the Tactical Environmental Support System
 Next Century [TESS(NC)] Meteorology and Oceanography
 (METOC) Database
- Unnumbered *Database Description for Digital Bathymetric Database –*
February 1997 *Variable Resolution (DBDB-V) Version 1.0*

2.2 Non-Government Documents

World Meteorological Organization, Geneva, Switzerland

- WMO-306 *Manual On Codes*
1995

3 SEGMENT OVERVIEW

The MADBV segment provides APIs used to access digitized bathymetry and bathymetric contour data in the TESS(NC) METOC Database. The schema for retrieving this data is defined by the data segment MDDBV. Both segments run in the DII COE release 3.1, hosted on the following machines and operating systems:

- Tactical Advanced Computer, TAC-3 (HP 750/755)/TAC-4, Operating System: HP-UX 10.20
- IBM-Compatible Personal Computer (PC), Operating System: Microsoft Windows NT 4.0, with Service Pack 3

Environmental Variable settings to find MADBV shared libraries are:

- NT Operating System
`setenv PATH=$PATH;c:/h/MADBV/bin`
- UNIX Operating System
`setenv SHLIB_PATH /h/MADBV/bin`

The MACRO definition to correctly set the NT declspec can be defined in one of three ways:

- Makefile using
`DEFINE = -D_MDBDLL`
- In code
`#ifdef _WIN32`
`#define -D_MDBDLL`
`#endif`
- Under project settings in the MS Visual Studio

The MADBV Segment is delivered as both archive and runtime libraries on both platforms, with filenames as follows:

<u>Library Type</u>	<u>HP-UX Filename</u>	<u>Windows NT Filename</u>
Archives	libMADBVAPI.a	MADBVAPI.lib
Runtime	libMADBVAPI.sl	MADBVAPI.dll

The runtime libraries are typically installed in the directory `/h/MADBV/bin` on HP systems and `C:\h\MADBV\bin` on Windows NT systems.

The archive libraries are typically installed in the directory `/h/MADBV/lib` on HP systems and `C:\h\MADBV\lib` on Windows NT systems.

The individual API methods are described in the API Reference Manual referenced in Section 2. Section 4 of this document provides instructions and programming examples for the use of the APIs.

4 SEGMENT DEVELOPMENT

Programming applications to access grid field data are straightforward. The Digitized Bathymetry API Segment provides interfaces to:

- Establish connection to the TESS(NC) MDDBV Database (MADBVConnect).
- Retrieve a listing of bathymetry resolutions available within the database (MADBVGetCatalog).
- Retrieve bathymetry along a line of bearing (MADBVGetTrack).
- Retrieve a grid of bathymetry data (MADBVGetGrid).
- Retrieve available bathymetric contours from the database (MADBVGetContours).
- Free the memory allocated for the contour data (MADBVFreeContours).
- Disconnect from the database (MADBVDisconnect).

Each of these methods is described in detail in the MADBV API Reference Manual (APIRM), referenced in Section 2. It is recommended that both the APIRM and this document be reviewed prior to programming with the APIs.

4.1 Writing Applications Using the MADBV APIs

This section shows the use of the MADBV APIs to perform common data access tasks. In each case, an overview of the actions to be performed is provided, along with a code example and a discussion of pertinent programming concerns.

In all cases, note that the database connect method (MADBVConnect) must be called to connect the application to the database before any other operations may be performed. The database disconnect method (MADBVDisconnect) should be called to disconnect the application from the database at the end of the session. These methods only need to be called once per session.

All structures must be initialized through use of calloc or memset. Failure to initialize a structure could cause unpredictable results. Uninitialized fields in an API data structure could cause a query to fail.

Section 4.2, MDDBV Database Overview, contains a description of the various data files accessed through these APIs.

The MADBVRET structure is used to return status information from each MADBV method. See Section 3.3.7 of the API Reference Manual for details of the information returned in this structure.

4.1.1 Retrieving Available Database Resolutions

The MADBVGetCatalog method is used to determine the resolutions available within the database. The retrieved catalog lists all resolutions available within the database, not the resolutions available for a specific area. This data may then be used to determine a sampling resolution for track or grid retrieval. It takes as input a pointer to a MADBVContext type (created at connection time), a pointer to float where the retrieved array of resolutions will be stored, and a pointer to type int where the number of available resolutions will be stored. It is the responsibility of the calling program to free the catalog array when it is no longer needed.

The code sample below demonstrates the calling interface for this method, as well as how to access the returned data. It also provides an example of utilizing the MADBVConnect and MADBVDisconnect APIs.

```
/*
*****
T E S T E R
*****
Purpose: Test driver for MADBVCatalog.
*****/
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MADBVAPI.h"
#include "MADBVErr.h"

void PrintCatalog(float *pCatArray, int NumFound);

/*
*****
M A I N
*****
*/
void main(argc, argv)
int argc;
char **argv;
{
    MADBVRET madbvRet;
    PMADBVCONTEXT pContext;
    int nNumFound;

```

```
float      *pCatArray;
long       lAccessFlag = 1;

printf("*****\n");
printf(" Testing the Catalog APIs:                \n");
printf(" MADBVCatalog                             \n\n");

madbvRet = MADBVConnect(&pContext, lAccessFlag);
if (madbvRet.nStatus)
{
    printf( "\tttester: Error calling MADBVConnect (%d) \n",
            madbvRet.nStatus);
}
else
{
    printf( "\tMADBVConnect : Successful!\n\n");

    /*****/
    /* Test the Catalog API */
    /*****/
    madbvRet = MADBVGetCatalog(pContext, &pCatArray, &nNumFound);

    /*****/
    /* Check for errors */
    /*****/
    if (madbvRet.nStatus)
    {
        printf( "\tMADBVCatalog: Error!(%d) (%s) \n",
                madbvRet.nStatus, madbvRet.szErrorMessage);
    }
    else
    {
        printf("\tMADBVCatalog: Successful !\n");
        PrintCatalog(pCatArray, nNumFound);
    }

    if (pCatArray) free(pCatArray);

    /*****/
    /* Call MADBVDisconnect */
    /*****/
    madbvRet = MADBVDisconnect(&pContext);
}

printf("Testing Complete                \n");
printf("*****\n");

exit( madbvRet.nStatus );

} /* End of main. */

void PrintCatalog(float *pCatArray, int NumFound)
{
```

```
int num = 0;
int i;

if (pCatArray == NULL )
{
    printf("\n===== NULL Catalog list.\n");
}

if (NumFound < 1)
{
    printf("\nNo resolutions were found.\n");
}
else
{
    printf("\n\nNumber of Resolutions found = %d ", NumFound);
}

printf("\n");
for (i=0; i<NumFound; i++)
    printf("%f\t", pCatArray[i]);

printf("\n\n");
}
```

4.1.2 Retrieving Bathymetry Along a Track

Bathymetry along a track (line of bearing) is provided by the MADBVGetTrack method. This method also provides bathymetry for a single point at a given latitude/longitude by specifying a track length of zero. MADBVGetTrack takes as input a pointer to an MADBVCONTEXT structure and a pointer to an MADBVTRACKQUERY structure. Additionally, a pointer to type int is passed in as the location to store the number of points retrieved, and the address of a pointer to type MADBVTRACKDATA is passed to be filled with the pointer to the retrieved data. This method will extract the data as specified in the MADBVTRACKQUERY structure, and return a pointer to an array of MADBVTRACKDATA structures, and the number of structures within the array. It is the responsibility of the calling program to free the MADBVTRACKDATA when it is no longer needed.

The PrintTrack routine demonstrates how to access the returned data, and the Prompt4Inputs routine demonstrates the format of the MADBVTRACKQUERY input structure and the format/limits/types of the inputs for the method.

An MADBVRET structure is returned.

```
/******
                                     T   E   S   T   E   R
```

Purpose: Test driver for MADBVGetTrack.

```
*****/
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MADBVAPI.h"
#include "MADBVErr.h"

void PrintTrack(int num_pts, PMADBVTRACKDATA pTrack);
void Prompt4Inputs(PMADBVTRACKQUERY pTrackQuery);

/*****
                                M   A   I   N
*****/
void main(argc, argv)
    int argc;
    char **argv;
{
    int          j = 0;
    MADBVRET     madbvRet;
    PMADBVCONTEXT pContext;
    int          nNumPoints;
    MADBVTRACKQUERY trackQuery, *pTrackQuery;
    PMADBVTRACKDATA pTrackData;
    long lAccessFlag = 0;

    pTrackQuery = &trackQuery;

    printf("*****\n");
    printf(" Testing the Track Retrieval APIs:           \n");
    printf(" MADBVGetTrack                                     \n\n");

    madbvRet = MADBVConnect(&pContext, lAccessFlag);
    if (madvbRet.nStatus)
    {
        printf( "\tttester: Error calling MADBVConnect (%d) \n",
                madbvRet.nStatus);
    }
    else
    {
        printf( "\tMADBVConnect : Successful!\n\n");

        Prompt4Inputs(pTrackQuery);

        /*****/
        /* Test the GetTrack API */
        /*****/
        madbvRet = MADBVGetTrack(pContext, pTrackQuery,
                                &nNumPoints, &pTrackData);

        /*****/
        /* Check for errors */
        /*****/
        if (madvbRet.nStatus)
        {
            printf("\tMADBVGetTrack: Error!(%d) (%s) \n",
                    madbvRet.nStatus, madbvRet.szErrorMessage);
        }
    }
}

```

```
    }
    else
    {
        printf("\tMADBVGetTrack: Successful !\n");
        PrintTrack(nNumPoints, pTrackData);
    }

    if (pTrackData != NULL) free(pTrackData);

    /******
    /* Call MADBVDisconnect */
    /******
    madbvRet = MADBVDisconnect(&pContext);

}

printf("Testing Complete                               \n");
printf("*****\n");

exit( madbvRet.nStatus );

} /* End of main. */

/*****
Other Functions
*****/
void Prompt4Inputs(PMADBVTRACKQUERY pQuery)
{
    char answer[128];
    float multiplier;
    char deg_txt[20], min_txt[20], sec_txt[20], hem[20];
    float degs, min, sec;
    int valid, extract_type, units;
    char land_check[20];

    valid = 0;
    while (!valid)
    {
        printf("\n\nEnter the starting latitude of the track (DD MM SS H): ");
        scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

        degs = atof(deg_txt);
        min = atof(min_txt);
        sec = atof(sec_txt);

        if (hem[0] != 's' && hem[0] != 'S' && hem[0] != 'n' && hem[0] != 'N')
            continue;

        if (degs > 90) continue;
        if (min > 60) continue;
        if (sec > 60) continue;

        valid = 1;
    }
    multiplier = 1;
    if (hem[0] == 's' || hem[0] == 'S') multiplier = -1;
    pQuery->rsLat = (degs + min/60.0 + sec/3600.0) * multiplier;
}
```

```
valid = 0;
while (!valid)
{
    printf("\n\nEnter the starting longitude of the track (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 'e' && hem[0] != 'E' && hem[0] != 'w' && hem[0] != 'W')
        continue;

    if (degs > 180) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 'w' || hem[0] == 'W') multiplier = -1;
pQuery->rsLon = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the extraction range (nm; max = 1000): ");
    scanf("%f", &pQuery->rsRange);

    if (pQuery->rsRange >= 0 && pQuery->rsRange <= 1000) valid = 1;
}

valid = 0;
while (!valid)
{
    printf("\n\nEnter the bearing: ");
    scanf("%f", &pQuery->rsBearing);

    if (fabs((double)pQuery->rsBearing) < 360.) valid = 1;
}

printf("\n\nExtraction type menu:\n");
printf("1- Specify extraction resolution in decimal minutes\n");
printf("2- Specify extraction resolution in nautical miles\n");
printf("3- Extract at the resolution of the database\n");

valid = 0;
while (!valid)
{
    printf("\nEnter the extraction type: ");
    scanf("%d",&extract_type);

    extract_type -= 1;
    if (extract_type >= 0 && extract_type <= 2)
        valid = 1;
}
pQuery->nExtractionType = extract_type;

if (extract_type == 0)
```

```
{
    valid = 0;
    while (!valid)
    {
        printf("\nEnter the extraction resolution (minutes; Max < 60): ");
        scanf("%f",&pQuery->rsResolution);

        if (pQuery->rsResolution >= 0 && pQuery->rsResolution < 60)
            valid = 1;
    }
}
else if (extract_type == 1)
{
    valid = 0;
    while (!valid)
    {
        printf("\nEnter the resolution (nm; Max = extract range): ");
        scanf("%f",&pQuery->rsResolution);

        if (pQuery->rsResolution >= 0 &&
            pQuery->rsResolution <= pQuery->rsRange)
            valid = 1;
    }
}

printf("\n\n Data unit menu:\n");
printf("1- Bathymetry in meters.\n");
printf("2- Bathymetry in feet.\n");
printf("3- Bathymetry in fathoms.\n");

valid = 0;
while (!valid)
{
    printf("\nEnter the data units: ");
    scanf("%d",&units);

    units -= 1;
    if (units >= 0 && units <= 2)
        valid = 1;
}
pQuery->nExtractionUnits = units;

valid = 0;
while (!valid)
{
    printf("Should the extraction end when land is encountered (y/n)? ");
    scanf("%s",land_check);

    if (land_check[0] == 'Y' || land_check[0] == 'y' ||
        land_check[0] == 'N' || land_check[0] == 'n')
        valid = 1;
}
pQuery->nLandCheckFlag = 1;
if (toupper(land_check[0]) == 'Y') pQuery->nLandCheckFlag = 0;
}

void PrintTrack(int num_pts, PMADBVTRACKDATA pTrack)
```

```
{
  int i;
  int deg[2];
  int min[2];
  float sec[2];
  char hem[2];
  double lat;
  double lon;

  if (!nDebug) return;

  for (i=0; i<num_pts; i++)
  {
    lat = fabs((double)pTrack->rsGCLatitude);
    deg[0] = (int)lat;
    lat -= deg[0]; lat *=60;
    min[0] = (int)lat;
    lat -= min[0]; lat *= 60;
    sec[0] = (float)lat;
    if (pTrack->rsGCLatitude < 0)
      hem[0] = 'S';
    else
      hem[0] = 'N';

    lon = fabs((double)pTrack->rsGCLongitude);
    deg[1] = (int)lon;
    lon -= deg[1]; lon *=60;
    min[1] = (int)lon;
    lon -= min[1]; lon *= 60;
    sec[1] = (float)lon;
    if (pTrack->rsGCLongitude < 0)
      hem[1] = 'W';
    else
      hem[1] = 'E';

    printf("%d %d %f %c   %d %d %f %c   %f nm %f\n",
           deg[0], min[0], sec[0], hem[0],
           deg[1], min[1], sec[1], hem[1],
           pTrack->rsRange, pTrack->rsBathymetry);

    pTrack++;
  }
}
```

4.1.3 Retrieving Gridded Bathymetry

The method MADBVGetGrid retrieves bathymetry at a specified resolution for a given area. Inputs to MADBVGetGrid include a pointer to the MADBVCONTEXT structure (created at connection time), a pointer to the MADBVGEOAREA structure that specifies the area boundaries, and a float containing the requested resolution (in decimal minutes). It also requires the address of integer values in which will be stored the number of rows and columns of bathymetry data retrieved, and the address of a pointer to float, where the pointer to the retrieved

array of bathymetry data (row/column ordering) will be stored. The retrieved bathymetry is returned in an array num_rows x num_columns of float, with the characteristic value of -10 used when bathymetry does not exist or the point is on land.

It is the responsibility of the calling routine to free the array of bathymetry data when it is no longer needed.

The status of the retrieval is returned by the method in an MADBVRET status structure.

All four routines return an MADBVRET return status structure.

```
/******  
T E S T E R  
Purpose: Test driver for MADBVGetGrid.  
*****/  
#include <math.h>  
#include <stdio.h>  
#include <time.h>  
#include "MADBVAPI.h"  
#include "MADBVErr.h"  
  
void PrintGrid(int numRows, int numCols, float *pBathymetry,  
              PMADBVGEOAREA pGeoArea, float rsResolution);  
void Prompt4Inputs(PMADBVGEOAREA pGeoArea, float *pResolution, int *pnUnits);  
  
/******  
M A I N  
*****/  
void main(argc, argv)  
    int argc;  
    char **argv;  
{  
  
    MADBVRET      madbvRet;  
    PMADBVCONTEXT pContext;  
    int           nNumPoints;  
    char          szPath_FileName[545];  
    MADBVGEOAREA geoArea, *pGeoArea;  
    float         *pBathymetry = NULL;  
    float         rsResolution;  
    int           nUnits;  
    int           numRows, numCols;  
    long          lAccessFlag = 1;  
  
    pGeoArea = &geoArea;  
  
    printf("*****\n");  
    printf(" Testing the Grid Retrieval APIs:          \n");  
    printf(" MADBVGetGrid                                \n\n");  
  
    madbvRet = MADBVConnect(&pContext, lAccessFlag);
```

```
if (madbvRet.nStatus)
{
    printf( "\tttester: Error calling MADBVConnect (%d) \n",
            madbvRet.nStatus);
}
else
{
    printf( "\tMADBVConnect : Successful!\n\n");

    Prompt4Inputs(pGeoArea, &rsResolution, &nUnits);

    /******
    /* Test the GetGrid API */
    /******
    madbvRet = MADBVGetGrid(pContext, pGeoArea, nUnits,
                            rsResolution, &numRows, &numCols,
                            &pBathymetry);

    /******
    /* Check for errors */
    /******
    if (madbvRet.nStatus)
    {
        printf( "\tMADBVGetGrid: Error!(%d) (%s) \n",
                madbvRet.nStatus, madbvRet.szErrorMessage);
    }
    else
    {
        printf("\tMADBVGetGrid: Successful !\n");
        PrintGrid(numRows, numCols, pBathymetry, pGeoArea, rsResolution);
    }

    if (pBathymetry != NULL);
    {
        free(pBathymetry);
        pBathymetry = NULL;
    }

    /******
    /* Call MADBVDisconnect */
    /******
    madbvRet = MADBVDisconnect(&pContext);

}

printf("Testing Complete \n");
printf("*****\n");

exit( madbvRet.nStatus );

} /* End of main. */

/******
Other Functions
*****/
void Prompt4Inputs(PMADBVGEOAREA pArea, float *prsResolution, int *pnUnits)
{
    char answer[128];
```

```
float multiplier;
char deg_txt[20], min_txt[20], sec_txt[20], hem[20];
float degs, min, sec;
int valid, extract_type, units;
char land_check[20];

valid = 0;
while (!valid)
{
    printf("\n\nEnter the latitude of the bottom (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 's' && hem[0] != 'S' && hem[0] != 'n' && hem[0] != 'N')
        continue;

    if (degs > 90) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 's' || hem[0] == 'S') multiplier = -1;
pArea->rsSLat = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the latitude of the top (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 's' && hem[0] != 'S' && hem[0] != 'n' && hem[0] != 'N')
        continue;

    if (degs > 90) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 's' || hem[0] == 'S') multiplier = -1;
pArea->rsNLat = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the longitude of the left side (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);
```

```
degs = atof(deg_txt);
min = atof(min_txt);
sec = atof(sec_txt);

if (hem[0] != 'e' && hem[0] != 'E' && hem[0] != 'w' && hem[0] != 'W')
    continue;

if (degs > 180) continue;
if (min > 60) continue;
if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 'w' || hem[0] == 'W') multiplier = -1;
pArea->rsWLon = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the longitude of the right side (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 'e' && hem[0] != 'E' && hem[0] != 'w' && hem[0] != 'W')
        continue;

    if (degs > 180) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 'w' || hem[0] == 'W') multiplier = -1;
pArea->rsELon = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the extraction resolution (minutes; Max < 60): ");
    scanf("%f", &prsResolution);

    if (*prsResolution >= 0 && *prsResolution < 60)
        valid = 1;
}

printf("\n\n Data unit menu:\n");
printf("1- Bathymetry in meters.\n");
printf("2- Bathymetry in feet.\n");
printf("3- Bathymetry in fathoms.\n");

valid = 0;
while (!valid)
{
    printf("\n\nEnter the data units: ");
```

```
scanf("%d",&units);

units -= 1;
if (units >= 0 && units <= 2)
    valid = 1;
}
*pnUnits = units;
}

void PrintGrid(int numRows, int numCols, float *pBathymetry,
              PMADBVGEOAREA pArea, float rsResolution)
{
    int i, j;

    printf("lower left corner is %f lat/ %f lon\n", pArea->rsSLat, pArea->rsWLon);
    printf("upper right corner is %f lat/ %f lon\n", pArea->rsNLat, pArea->rsELon);
    printf("resolution is %f\n\n",rsResolution);

    for (i=0; i<numRows; i++)
    {
        for (j=0; j<numCols; j++)
            printf("%10.4f ",pBathymetry[i*numCols + j]);

        printf("\n");
    }
}
```

4.1.4 Retrieve Bathymetry Contours

The MABVGetContours method retrieves bathymetry contours generated from the digitized bathymetry database. MADBVGetContours requires a pointer to an MADBVGEOAREA structure defining the area for which contours are to be extracted, the address of a pointer to an MADBVCONTOURDESC structure where the pointer to the contour data will be stored, and a pointer to the integer location where the number of contour levels retrieved will be stored.

The contour data is freed with a call to MADBVFreeContours, which is also demonstrated in the code example below. Note that the contour data routines do not require that a connection to the databases exists, and therefore do not require a database context as an argument. The contour database, while a subset of the MDDBV segment data, is not part of the OAML DBDB-V database.

/*****

T E S T E R

Purpose: Test driver for MADBVGetContours.

*****/

```
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "MADBVAPI.h"
#include "MADBVErr.h"
```

```
void WriteContourFile(PMADBVCONTOURDESC pContour, int NumContours);
void Prompt4Inputs(PMADBVGEOAREA pGeoArea);
```

/*

M A I N

*/

```
void main(argc, argv)
    int argc;
    char **argv;
{
```

```
    MADBVRET      madbvRet;
    PMADBVCONTOURDESC pContour;
    int           nNumContours;
    MADBVGEOAREA  geoArea, *pGeoArea;
```

```
    pGeoArea = &geoArea;
```

```
    printf("*****\n");
    printf(" Testing the Contour Retrieval APIs:          \n");
    printf(" MADBVGetContours          \n\n");
```

```
    Prompt4Inputs(pGeoArea);
```

```
    /* Test the GetContours API */
```

```
    madbvRet = MADBVGetContours(pGeoArea, &nNumContours, &pContour);
```

```
    /* Check for errors */
```

```
    if (madvbRet.nStatus)
    {
        printf( "\tMADBVGetContours: Error!(%d) (%s) \n",
                madbvRet.nStatus, madbvRet.szErrorMessage);
    }
```

```
    else
    {
        printf("\tMADBVGetContours: Successful !\n");
        WriteContourFile(pContour, nNumContours);
```

```
        madbvRet = MADBVFreeContours(nNumContours, pContour);
```

```
    if (madbvRet.nStatus)
    {
        printf("\tMADBVGetContours: Error!(%d) (%s) \n",
            madbvRet.nStatus, madbvRet.szErrorMessage);
    }
    else
        printf("\tMADBVFreeContours successful\n");
}

printf("Testing Complete \n");
printf("*****\n");

exit( madbvRet.nStatus );

} /* End of main. */

/*****
                                Other Functions
*****/
void Prompt4Inputs(PMADBVGEOAREA pArea)
{
    char answer[128];
    float multiplier;
    char deg_txt[20], min_txt[20], sec_txt[20], hem[20];
    float degs, min, sec;
    int valid, extract_type, units;
    char land_check[20];

    valid = 0;
    while (!valid)
    {
        printf("\n\nEnter the latitude of the bottom (DD MM SS H): ");
        scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

        degs = atof(deg_txt);
        min = atof(min_txt);
        sec = atof(sec_txt);

        if (hem[0] != 's' && hem[0] != 'S' && hem[0] != 'n' && hem[0] != 'N')
            continue;

        if (degs > 90) continue;
        if (min > 60) continue;
        if (sec > 60) continue;

        valid = 1;
    }
    multiplier = 1;
    if (hem[0] == 's' || hem[0] == 'S') multiplier = -1;
    pArea->rsSLat = (degs + min/60.0 + sec/3600.0) * multiplier;

    valid = 0;
    while (!valid)
    {
        printf("\n\nEnter the latitude of the top (DD MM SS H): ");
        scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

        degs = atof(deg_txt);
```

```
min = atof(min_txt);
sec = atof(sec_txt);

if (hem[0] != 's' && hem[0] != 'S' && hem[0] != 'n' && hem[0] != 'N')
    continue;

if (degs > 90) continue;
if (min > 60) continue;
if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 's' || hem[0] == 'S') multiplier = -1;
pArea->rsNLat = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the longitude of the left side (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 'e' && hem[0] != 'E' && hem[0] != 'w' && hem[0] != 'W')
        continue;

    if (degs > 180) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
multiplier = 1;
if (hem[0] == 'w' || hem[0] == 'W') multiplier = -1;
pArea->rsWLon = (degs + min/60.0 + sec/3600.0) * multiplier;

valid = 0;
while (!valid)
{
    printf("\n\nEnter the longitude of the right side (DD MM SS H): ");
    scanf("%s %s %s %s", &deg_txt, &min_txt, &sec_txt, &hem);

    degs = atof(deg_txt);
    min = atof(min_txt);
    sec = atof(sec_txt);

    if (hem[0] != 'e' && hem[0] != 'E' && hem[0] != 'w' && hem[0] != 'W')
        continue;

    if (degs > 180) continue;
    if (min > 60) continue;
    if (sec > 60) continue;

    valid = 1;
}
}
```

```
    multiplier = 1;
    if (hem[0] == 'w' || hem[0] == 'W') multiplier = -1;
    pArea->rsELon = (degs + min/60.0 + sec/3600.0) * multiplier;
}

void WriteContourFile(PMADBVCOUTOURDESC pContour, int NumContours)
{
    int i, j;
    static int counter = 0;
    char filename[128];
    FILE *fp;
    PMADBVCOUTOURDESC pDescrip;

    sprintf(filename, "contour%d.dat", counter++);
    fp = fopen(filename, "wb");
    pDescrip = pContour;

    fwrite(&NumContours, sizeof(int), 1, fp);

    for (i=0; i < NumContours; i++)
    {
        fwrite(&pDescrip->depth, sizeof(int), 1, fp);
        fwrite(&pDescrip->numSeg, sizeof(int), 1, fp);
        pDescrip++;
    }

    pDescrip = pContour;
    for (i=0; i < NumContours; i++)
    {
        fwrite(pDescrip->pContour, sizeof(MADBVCNTR_LINE_STRUCT),
            pDescrip->numSeg, fp);
        pDescrip++;
    }

    fclose(fp);
}
```

4.2 MDDBV Database Overview

The Digitized Bathymetric Data Segment consists of a static dataset developed by NAVOCEANO, and a pair of bathymetry contour files generated from the NAVOCEANO dataset. The Ocean Floor Depth DBDB-V is a digital bathymetric database that provides ocean depths at various gridded resolutions.

The NAVOCEANO-developed dataset for DBDB-V consists of four file types. The depth information is expressed in meters, uncorrected at an assumed sound velocity of 1500 m/s. The file types are:

- **Master Index File** – Contains a pointer or byte address to each populated 1-deg cell for each of the available resolutions.
- **Index File** – Provides a link to the detailed depth values, as well as a link to a description file associated with the depths.
- **Description File** – Provides details on the compressions, scaling, and storage of the depth information.
- **Data File** – contains the depth values for a 1-deg cell of a specific resolution.

The data file formats can be found in the DBDB-V Database Description Document (see Section 2.1).

Two additional data files have been generated using the DBDB-V data in order to provide bathymetry contours suitable for display on a chart product. The contour data is a static global bathymetry database consisting of non-overlapping tiles. The tile size (degrees on a side) is specified in the contour data index file (word 2 of the header). Tiles for areas South of 78 S were not generated, so the indexing starts at 78 S, 0 E. The file types are:

- **Index File** – Contains the number of contour levels available, and for each tile, the details each contour level and the address of the contour data in the contour data file. Also contains header data indicating how many contour levels exist in the database, as well as the size of the contour tiles.
- **Contour Data File** – Contains the contour segments for each contour level.

The format of the contour data files is given in Appendix A.

This DII COE-compliant Data Segment must be installed on a file system that can be mounted by the platform running the applications that require this data. Unlike DII COE database segments, data segments are accessed directly by the associated APIs. This requires that the Data Segment be visible to the application accessing it. This visibility is accomplished by means of an NFS mount of the volume containing the data segment (Figure 1-3). If an application utilizing the MDDBV segment resides on a different machine than the database, access to the database is available via NFS services if the drive is mounted.

4.3 Building Applications Using the MADBV Libraries

This section contains four makefiles that can be used to build the MADBV APIs. Static and dynamic makefiles are provided for the HP-UX and the Windows NT environments. The dynamic makefiles use the run-time libraries (*.sl) on HP-UX or (*.dll) on Windows NT to build the APIs. The static makefiles use the static libraries (*.a) on HP-UX, (*.lib) on Windows NT to build the APIs.

4.3.1 Makefile Using the Dynamic/Static MADBV Libraries on HP-UX

```
### Dynamically Linked Program
PROGRAM1 = MADBV_Ingest_d

### Statically Linked Program
PROGRAM2 = MADBV_Ingest_l

CFILE = main.c

OFILE = $(CFILE:.c=.o)

RM = /bin/rm -f

LIBS = -lMADBVAPI

INCLUDES = -I$(MADBV_HOME)/include

all : $(PROGRAM)

CFLAGS = $(INCLUDES) -Aa

# Dynamic Linked Program
$(PROGRAM1) : $(OFILE)
    $(CC) $(OFILE) \
    -L$(MADBV_HOME)/bin $(LIBS) \
    -lc -lm -o $(PROGRAM1)

# Dynamic Linked Program
$(PROGRAM2) : $(OFILE)
    $(CC) $(OFILE) \
    -L$(MADBV_HOME)/lib $(LIBS) \
    -lc -lm -o $(PROGRAM2)

clean :
    $(RM) $(OFILE) $(PROGRAM1) $(PROGRAM2) core
```

4.3.2 Makefile Using the Dynamic MADBV Libraries on Windows NT

```
PROGRAM = MADBV_Store

CFILE = main.c

OFILE = $(CFILE:.c=.obj)
```

```
MADBV_HOME = ..\..\..\..\..
RM = -@del /Q /F /S
LD = link

LIBS = $(MADBV_HOME)\bin\MADBVAPI.lib

INCLUDES = -I$(MADBV_HOME)\include

all : $(PROGRAM)

CFLAGS = $(INCLUDES) -MD

$(PROGRAM) : $(OFFILE)
    $(LD) $(OFFILE) \
    $(LIBS) -OUT:$@

clean :
    $(RM) $(OFFILE) $(PROGRAM)
```

4.3.3 Makefile Using the Static MADBV Libraries on Windows NT

```
PROGRAM = MADBV_Store

CFILE = main.c

OFFILE = $(CFILE:.c=.obj)

MADBV_HOME = ..\..\..\..\..
RM = -@del /Q /F /S
LD = link

LIBS = $(MADBV_HOME)\lib\MADBVAPI.lib

INCLUDES = -I$(MADBV_HOME)\include

all : $(PROGRAM)

CFLAGS = $(INCLUDES)

$(PROGRAM) : $(OFFILE)
    $(LD) $(OFFILE) /NODEFAULTLIB:LIBC.LIB /NODEFAULTLIB:MSVCRT.LIB\
    $(LIBS) -OUT:$@

clean :
    $(RM) $(OFFILE) $(PROGRAM)
```

(This page intentionally left blank.)

5 CUSTOMIZING SEGMENTS

This section is tailored out.

(This page intentionally left blank.)

6 NOTES

6.1 Glossary of Acronyms

AESS	Allied Environmental Support System
API	Application Program Interface
APIRM	API Reference Manual
COE	Common Operating Environment
COTS	Commercial Off-the-Shelf
DBDB-V	Digitized Bathymetric Database – Variable Resolution
DBDD	Database Design Description
DII	Defense Information Infrastructure
GCCS	Global Command and Control System
IC4ISR	Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance
IMOSS	Interim Mobile Oceanographic Support System
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MADBV	Digitized Bathymetry (Variable Resolution) API Segment of the TESS(NC) METOC Database
MDDBV	Digitized Bathymetry (Variable Resolution) Database Segment of the

	TESS(NC) METOC Database
MDGRID	Grid Field Database Segment of the TESS(NC) METOC Database
MDIMG	Imagery Database Segment of the TESS(NC) METOC Database
MDLLT	LLT Observations Database Segment of the TESS(NC) METOC Database
MDREM	Remotely Sensed Data Database Segment of the TESS(NC) METOC Database
MDTXT	Textual Observations Database Segment of the TESS(NC) METOC Database
METOC	Meteorology and Oceanography
MIDDS	Meteorological Integrated Data Display System
NFS	DII COE Kernel Service
NITES	Navy Integrated Tactical Environmental Subsystem
OAML	Oceanographic and Atmospheric Master Library
PM	Programming Manual
PS	Performance Specification
RDBMS	Relational Database Management System
SQL	Structured Query Language
TESS(NC)	Tactical Environmental Support System Next Century

Appendix A - Contour Data File Formats

The contour data portion of the MDDBV Data Segment consists of two files:

Index File (dbdbvcon.ndx)

The index file contains a two word header organized as follows:

```
int    num_contours; /* number of contour levels generated */
int    tile_size;    /* size in decimal degrees per side of contour data tiles */
```

The remainder of the index file contains contour level description data for each tile within the contour data file. The first record corresponds to the tile whose lower left corner is at 78 S, 0 E (data does not exist below 78 S). Each record contains the following data:

```
DBDBC_CONTOUR contour_data[num_contours]; /* contour level descriptions
                                           (num_contours from header */
long offset; /* offset into contour data file for this tile */
```

where DBDBC_CONTOUR has the following format:

```
struct dbdbc_contour
{
    int    depth; /* depth (fathoms) of contour level */
    int    num_seg; /* number of contour segments for this depth */
};
typedef struct dbdbc_contour DBDBC_CONTOUR;
```

If the given offset is -1, no contour data exists for this tile.

Data File (dbdbv.con)

The data file consists of records of the following structure:

```
{
    float  lat[2]; /* starting and ending latitudes for contour segment */
    float  lon[2]; /* starting and ending longitude for contour segment */
};
```

For each tile, the data is organized in the order of the index information. i.e. for the first DBDBC_CONTOUR array element, there will be DBDBC_CONTOUR[0]->num_seg contour segments corresponding to depth DBDBC_CONTOUR[0]->depth.

(This page intentionally left blank.)